Shell Programming is designed to be used to:

* Run several commands together
  + Putting multiple commands in a single file so you can frequently use them, instead of repeat typing out

On a LINUX / UNIX machine, adding “**#!/bin/sh**” to all your shell script files is necessary.

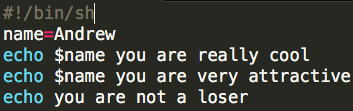
* This tells your operating system the correct interpreter to use to run your file.
* If you aren’t writing shell script, but python, perl etc. then you would need **#!/bin/perl** or the equivalent

When executing script files, you need to first run the command “**chmod +x <filename>**”.

* This command changes permission of your file and makes it executable.
* After changing permissions, you can run the script as as normal program with “**./**”

Shell variables example

* No need to declare type of shell variables, simply use them. E.g. x=5
* Use the value of variable by preceding the variable name with a “**$**” sign:



* Changing the value of the variable e.g. to “Hermione” would change the output too.

Shell doesn’t give error messages, so if you **“echo $stupid”** / a non-existent variable, it will just output an empty line.

**$** is a Shell meta-character.

* To use the actual “$” symbol, you need to use a **“\” (backslash)**, telling shell to “ignore any special meaning of the following character”
* You can also use **‘single quotations’** to do the same thing

**Shell file matching special characters**

**\*** **(asterisk)** in Shell looks for files that match it as a pattern.

* Example – Matching file types: 
* Corresponding Regular Expression = “ **.\*** ” **(dot-star)** that does pattern matching, where the asterisk is any number or character.

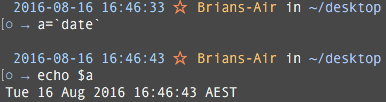
**?** **(qn mark**) in Shell does file matching with ONE CHARACTER.

* Example 1 – Matching any file type with two characters: 
* Example 2 – Matching other file types: 

**[ ] (square brackets)** in Shell does file matching with ONE CHAR FROM THIS LIST

* Example – Matching C type files or H type files: 

**`string` (backquotes)** in Shell will execute whatever command that is between the backquotes, get the output and put it where the backquotes are.

* Basically, you can get the whole output of a command as a string.
* Example – Echo date command using backquotes:

If this was done regularly without backquotes,  
the echo would = date , rather than the output  
of the command date.

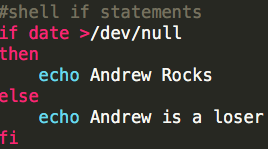
**Expr (Shell and Arithmetic):**

* For doing maths in shell, run a program **expr**
* Example 1 – Addition: 
* Example 2 – Multiplication: 
  + make sure to use **\** (backslash) or **‘ \* ‘** (single quotations) to tell Shell to ignore **\*** as a special characters
* EXPR cannot be used for large computations, otherwise it will give an overflow error.

**/dev/null** is a special “black hole” file on LINUX / UNIX.

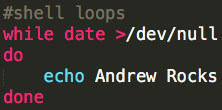
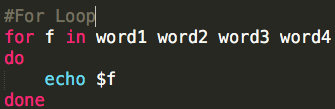
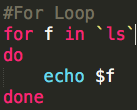
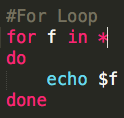
* You can write all you like to dev/null and nothing happens
* You can read all you like to dev/null and it shows 0 bytes
* Generally, if there is output that you don’t want, send it off to dev/null
* This saves you time adding an option to every program to tell them “not to show any output”

**If statements** in Shell:

* Example:

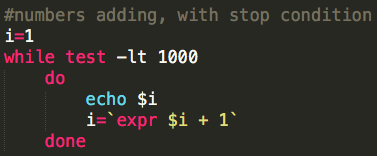
Output = Andrew Rocks  
Date output would not show because of /dev/null

**Loops** in Shell:

* WHILE LOOP Example:  
    
  Output = Andrew Rocks loop
* FOR LOOP Example:  
  “For variable in list of strings,  
  do commands, done”  
    
  Output = executes once per string / each line is a filename

**Sleep** in Shell

* Tells the cpu to “wait for X seconds” after running the program
* Useful for loops, if you want to be able to read things as it loops.
* Example: **while date**  
   **echo “Andrew Rocks”  
   sleep 2  
   done**

Using **Test** command in Shell to stop programs

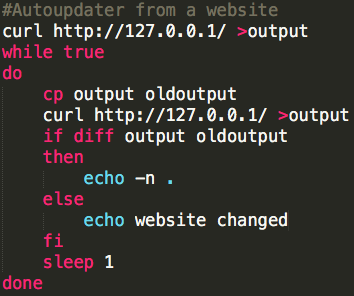
* Example: Stopping a loop

**test –lt 1000**  
means  
“while number is less than 1,000”

**$#** **(dollar-hash)** in Shell stores the number of command line arguments.

Shell Debugging

* Get the shell to tell you everything that it executes:
  + In the command line: **sh –x**
  + In a script: **set –x**
* Useful to do inside a script, as you can place the debugging command up to a point where you know everything is working. You will be able to avoid seeing the output for the first / correct part of the script.

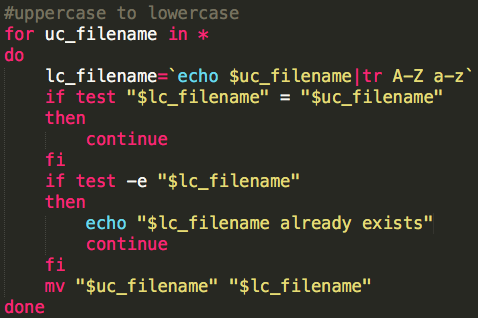
**ADD NOTES ABOUT BREAK / CONTINUE**

**wget** in Shell saves the website to a file.

* **Curl** is a similar tool that transfers data from or to a server.
* Example: Checking for changes on a website

Input:  
**While output == oldoutput,  
 echo . . .  
else  
 echo website changed.**

Changing uppercase to lowercase filename



Having double quotes around **“$lc\_filename”** / **“$uc\_filename”** are important  
 as spaces can cause problems.  
  
 Usually, you don’t know the type of data you are taking from a website and  
 you can’t be 100% that there are no spaces in names etc. so it is best practise  
 if you use double quotes when using variables.